Administration of Broker Stubs

This chapter covers the following topics:

- Available Stubs
- Linking the Stubs
- Transport Methods for Broker Stubs
- Using Job Variables
- Using BROKER under openUTM

Available Stubs

This table lists all Broker stubs available under BS2000/OSD that are to be used with the programming languages Assembler | C | COBOL | Natural | PL/I.

Note:
Use of the transport method NET will greatly improve performance when running Broker kernel and applications on the same machine. We recommend using the transport method NET for all local communication within BS2000/OSD. In order to use the transport method NET for messages involving more than 32 KB, you must install Adabas 8.1 cross-memory services. If you have not yet installed Adabas 8.1 cross-memory services, you can instead use TCP/IP to transport more than 32 KB of data.

Note for Adabas 8.1 users

When using Adabas 8.1.1 with any of the BS2000/OSD stubs to transport more than 32 KB of data, note the following:

- Adabas/WAL 8.1 must be installed.
- The Adabas/WAL 8.1 link routine must be used by the application or TP monitor.
- Adabas/WAL 8.1 libraries must be used by the Broker kernel.
- Adabas/WAL 8.1 libraries must be used by the Broker stubs.
- The parameter `EXTENDED-ACB-SUPPORT` must be used for transmitting data from Adabas (NET).
- Sufficient buffer space by `IUBL`, `NABS` and `NUM-COMBUF` must be specified.

The following stubs are available:

Name	Environment	Supported Transport Method
BKIMBTIA	All environments that use batch or Dialog (formerly TIAM)	Adabas communication, SSL and TCP/IP
BROKER	All environments that use batch or Dialog (formerly TIAM)	Adabas communication, SSL and TCP/IP

Note:
`BKIMBTIA` was dropped after release 8.1. Stub `BROKER` provides all features of `BKIMBTIA`.

Linking the Stubs

This section covers instruction for linking stubs:

- Stub BROKER
- Stub BROKER with Natural

Stub BROKER

To prepare your application to perform Broker calls

1. Link the front-end module `BROKER` from the EntireX load library (EXX811.LIB) to your application. It has the entry point "`BROKER`". When `BROKER` is first called, it loads the actual stub module from the EntireX load library and transfers control to it.
2. Add the following assignment to the startup procedure:

   ```
   /ADD-FILE-LINK LINK-NAME=ETBLIB,FILE-NAME=<EXX_load_library>
   ```

3. To enable the Adabas transport method, add the following assignment to the startup procedure:

   ```
   /ADD-FILE-LINK LINK-NAME=DDLIB,FILE-NAME=<adabas_load_library>
   ```

   As a result, the required Adabas link module is loaded from the appropriate Adabas load library.

4. Add the following assignment to the startup procedure:

   ```
   /ADD-FILE-LINK LINK-NAME=DDLNKPAR,FILE-NAME=<adalnk-parameter>
   ```

   As a result, ADAUSER reads the configuration parameters, for example IDTNAME.

Stub BROKER with Natural

To prepare your application to perform Broker calls

1. Add the following assignment to the startup procedure:

   ```
   /ADD-FILE-LINK LINK-NAME=BLSLIB00,FILE-NAME=<EXX_load_library>
   ```

2. Add the following assignment to the startup procedure:

```
/ADD-FILE-LINK LINK-NAME=ETBLIB,FILE-NAME=<EXX_load_library>
```

3. Start Natural with the following profile parameters:

   ```
   RCA=(BROKER),RCALIAS=(BROKER,BROKER)
   ```

   As a result, BROKER is loaded dynamically, and each broker call will use this stub.

   Note:
   This dynamic load/execute will work even if an old NATETB23 has already been linked to the shared Natural nucleus as static module. You need not link BROKER statically to the Natural front-end. It is, however, possible to link BROKER statically to the front-end Natural and remove the NATETB23 module from the shared Natural nucleus to avoid specifying the profile parameters mentioned above.

Transport Methods for Broker Stubs

This section covers the following topics:

- Transport Method Values
- Default Transport Methods
- Using Transport Methods
- Setting the Timeout for the Transport Method
- Tracing for Broker Stubs

Transport Method Values

The following table describes the possible values for the transport methods:

<table>
<tr><th>Transport Value</th><th>Description / Tips</th></tr>
<tr><td>NET</td><td>Use Adabas BS2000/OSD Communication Environment as transport method. It is also possible to communicate remotely with the transport method NET from an application (client or server) to the broker kernel using Entire Net-Work. For remote NET communication, Entire Net-Work must be installed both on the machine where the broker kernel runs and on the machine where your application (client or server) runs, and a connection must be established.

Using Adabas/WAL V8 allows more than 32 KB of data to be communicated. Otherwise the following maximum values are allowed:

ACI Version — Max Send/Receive length
1 — 32167
2, 3 — 31647
4-8 — 31643
9 or above — 31123

Note:
If Adabas version 8 is not used, these same limits still apply under BS2000/OSD.</td></tr>
<tr><td>TCP</td><td>Use TCP/IP as transport method.</td></tr>
</table>

Default Transport Methods

Stub	Default Transport Method
BKIMBTIA	NET
BROKER	NET

Using Transport Methods

This section covers specifications for transport methods as part of the broker ID.

Note:
If no transport method has been specified as part of the broker ID, default value NET is used.

- **Using Adabas Communication**

  To Use Adabas Communication as Transport Method

  - Specify:

    ```
    broker-id::NET
    ```

    Notes:

    1. Port number does not apply and is therefore left blank. Adabas communication is the transport method.
    2. It is not possible to provide the `IDTNAME` with the broker ID. The `IDTNAME` is specified in a parameter file controlled by the `ADAUSER` module (assigned using link name `DDLNKPAR`).
- **Using TCP/IP**

  To use TCP/IP as transport method

  - Specify:

    ```
    broker-id:nnnnn:TCP
    ```

    where *nnnnn* is a placeholder for a port number.

Setting the Timeout for the Transport Method

Introduction

If the transport layer is interrupted, communication between the broker and the stub - that is, client or server application - is no longer possible. A client or server might possibly wait infinitely for a broker reply or message in such a situation. To prevent this and return control to your calling application in such a situation, set a timeout value for the transport method.

The timeout settings for transport layers are independent of the timeout settings of the broker.

Setting the timeout for the transport layer is possible for the transport method TCP, and is supported by broker stub `BROKER`.

Transport Timeout Values

The timeout value for the transport method is set by the environment variable `ETB_TIMEOUT` on the stub side. This transport timeout is used together with the broker timeout - which is set by the application in the `WAIT` field of the broker ACI control block - to calculate the actual value for the transport layer's timeout. The following table describes the possible values for the transport timeout:

Transport Timeout Value	Description
0	Infinite wait for the application.
n	The transport method additionally waits this time in seconds. A negative value is treated as `TIMEOUT=0` (infinite wait for the application).
nothing set	Transport method waits additional 20 seconds.

The actual timeout for transport layer equals broker timeout (`WAIT` field) + timeout value for transport method.

Tracing for Broker Stubs

Scope

Setting tracing is supported by the broker stub `BROKER` if transport method TCP is used. The stub tries to access the SDF variable `ETB-STUBLOG` (or, failing that, a job variable with the same name), to evaluate the value of the logging level. If the logging level is set, a sequential file will be created with the file name *9999.ETB* where *9999* is the task sequence number of the running task.

Trace Level		Description
0	`NONE`	No tracing. Switch tracing off.
1	`STANDARD`	Traces initialization, errors, and all ACI request/reply strings.
2	`ADVANCED`	Used primarily by system engineers, traces everything from level 1 and provides additional information, for example the Broker ACI control block, as well as information from the transports.
3	`SUPPORT`	This is full tracing through the stub, including detailed traces of control blocks, message information, etc.

Activating Logging

To activate logging

- Set `JV ETB-STUBLOG` to value [1|2|3].

Where "1" is the lowest log level and "3" is the highest.

Using Job Variables

SDF and job variables (environment variables in an Open Systems architecture, i.e. UNIX or Windows) are used with the stub BROKER to read configuration parameters. BS2000/OSD uses the hyphen character whereas environment variables use the underscore character. The stub attempts to read the SDF variable. If this fails, the job variable is read. If neither an SDF variable nor a job variable is read, it is assumed not using any environment variables.

Using BROKER under openUTM

You cannot use BROKER with dialog transactions under openUTM. You can, however, use BROKER within asynchronous transaction processing under openUTM. Prepare your Natural/UTM application as follows:

1. Link module BROKER from the EntireX library *EXX811.LIB* to the front-end part of your Natural/UTM application.
2. Add the following assignment to the Natural/UTM startup job:

   ```
   /ADD-FILE-LINK LINK-NAME=ETBLIB,FILE-NAME=EXX_load_library
   ```

3. To enable the Adabas transport method, add the following assignment to the Natural/UTM startup job:

   ```
   /ADD-FILE-LINK LINK-NAME=DDLIB,FILE-NAME=adabas_load_library
   ```

For more information on writing an asynchronous Natural/UTM transaction see section *Asynchronous Transaction Processing under UTM* in the Natural/UTM documentation.